Notes on View Synchronization using Default Logic
(Extended Abstract)

Giuseppe Polese and Mario Vacca

Università di Salerno, Dipartimento di Matematica e Informatica
Via Ponte don Melillo, 84084 Fisciano (SA), Italy
{gpolese, mvacca}@unisa.it

Abstract. The synchronization of views is one of the schema evolution problems and it calls for the redefinition of those views becoming undefined after a schema change, in order to keep them still working on the new schema. This problem is particularly difficult for capacity reducing schema changes, when it could be only possible to approximate the existing views. Recently, the use of schema mappings to express schema changes has allowed both to deal with a wide range of schema change operations and to facilitate the view synchronization; but approximating views requires mappings able to describe approximate schema changes. This paper introduces the default schema mappings, a new kind of mappings based on default logic and it provides a preliminary study showing the possibility of using them to realize an approximate view synchronization process.

Key words: Default schema mapping, default query, schema evolution, schema mapping, view synchronization

1 Introduction

Database schema evolution naturally occurs during the life cycle of an information system: modifications of the schema can take place due to either changes in the world, or errors, or design decisions, and they can yield very significant problems. The view synchronization problem, a very challenging one in the context of data warehouses [1], has been increasing its importance with the development of the Semantic Web [2], becoming one of main issues of schema evolution: it arises when, after a schema change, some of the existing queries and views stop working, and it is necessary to redefine them over the new schema.

The view synchronization problem was first addressed, defined, named and classified by Rundensteiner, Lee and Nica, who also proposed a solution to it [1]. This problem is particularly difficult when the schema evolution causes the loss of some information, as it happens in the case of capacity reducing schema changes, like the deletion of attributes: some views, defined on a given schema, might not be reformulated in an exact way on the evolved one [3,1,4]. This problem differs
from the query reformulation one, as it requires more kinds of relations between the old and the new (synchronized) view extents to be considered [1].

Many solutions have been proposed for the problem of view synchronization: in [5] only a limited number of schema change operations (including capacity reducing ones) were taken in account, because this approach needs different algorithms for each change operation; a recent approach [4] aiming at building a tool for schema evolution, has broadened the set of change operations allowed, defining their semantics by schema mappings. In general, the Generic Model Management based approaches (see, for example, [6]), expressing changes by mappings, are able to deal with a wide range of schema change operations, but they don't solve the problem in the case of capacity reducing schema changes. Therefore, the need arises for mappings and operators allowing to deal with capacity reducing schema changes.

This paper describes the preliminary results of an on-going research project whose aim, to quote Lakshmanan et al., is to try to answer the question: “How can we produce meaningful answers to queries (based on an older version of the schema) which refer to such “lost” information?” [3]. A first step toward the realization of this project is the introduction of suitable mappings for dealing with capacity reducing schema change operations. Our idea is the following: mappings are a way to express relations between schemas and they are defined in terms of relations between instances; mappings between two schemas $S$ and $T$ are described by “set of formulas of some logical formalism over $(S,T)$” (Fagin et al. [8], p. 999). In cases like the deletion of attributes, when trying to “recover” the data of the source schema, we could accept to recover almost all the data, allowing some errors. Therefore, the mapping should carry exactly almost all the data or as many data as possible. To this end, it is possible to use mappings expressed by approximate laws. In general, a way to approximate laws is just “weakening” them and this can be made considering general laws [9]. The default logic [10] is a suitable logical formalism to express general laws and, moreover, it has been applied to databases [9] and a query language based on it (DQL) has been proposed [11].

In this paper, we show that it is possible to define a new kind of mappings, based on the default logic, that we call default schema mapping, which is suitable for dealing with cases in which information is lost, and that these mappings can be used to realize an approximate view synchronization process.

---

1 See “The EVE Project” http://davis.epi.edu/~dsrg/PROJECTS/EVE/index.html for details and references.
3 More precisely, in this case it was proposed to use a non-generic model semantics: in fact “...we can write a function $f$ that encapsulates the semantic knowledge necessary to strip out parts of a view definition .... Thus, $f$ gives us a way of exploiting non-generic model semantics while still working within the framework of the model management algebra.” [7]
2 Motivating example

In general, the problem of view synchronization can be stated in the following way: when a schema changes, some of its views and queries can become undefined; the problem is how to “fix” their definitions so that they can continue to work on the new schema.

Example 1. Consider the schema \( S = \{ \text{Cities}(\text{Name}, \text{Country}, \text{Pop}, \text{Capital}) \} \) storing information about cities of a country and the view \( \text{IsCapital}(x, y) \iff \text{Cities}(x, y, z, t) \land t = \text{"yes"} \) of the capitals of the countries. Consider the schema \( T = \{ \text{Cities}'(\text{Name}, \text{Country}, \text{Pop}) \} \) obtained from \( S \) deleting the attribute \( \text{Capital} \) from the relation \( \text{Cities} \). The passage from the schema \( S \) to the schema \( T \) can be described by the projection \( \forall x, y, z, t(\text{Cities}(x, y, z, t) \rightarrow \text{Cities}'(x, y, z)) \). Synchronizing \( \text{IsCapital} \) means trying to rewrite it using the information from the schema \( T \).

When trying to fix a view definition, the best result you could expect is to rewrite it in terms of the new schema, so that the new view has the same extent as the old one. Unfortunately, this is not always possible [3], and you have two possibilities: either to drop the view or to be satisfied with a view whose extent is close to that of the original one. In the following sections we show that, using the default logic, it is possible to approximate the process of view synchronization.

3 Default logic and default queries

In this section we recall some basic concepts of default logic and default queries; for further details see [10,11].

The basic concept of default logic [10] is the default rule, a formula of the form

\[
\frac{\alpha(x); \beta_1(x), \ldots, \beta_n(x)}{\gamma(x)}
\]

where \( \alpha(x) \) is the prerequisite of the default, \( \beta_1(x), \ldots, \beta_n(x) \) \((n \geq 0)\) are the justifications and \( \gamma(x) \) is the consequence.

A default theory is a pair \( \langle B, D \rangle \), where \( B \) is a set of closed first-order formulas and \( D \) is a set of default rules.

The semantics of a default theory is based on the concept of extension, defined as follows (\( Cons() \) is the deductive closure):

\[
E_0 = B
\]

\[
E_{i+1} = Cons(E_i) \cup \left\{ \gamma \mid \alpha \beta_1 \ldots \beta_n \in D, \alpha \in E_i, \neg \beta_j \not\in E \text{ for all } j \right\}
\]

A formula \( \phi \) is entailed from a default theory \( \langle B, D \rangle \) under the credulous semantics (resp. skeptical/cautious semantics) if \( \phi \) is in at least one extension of \( \langle B, D \rangle \) (resp. all the extensions of \( \langle B, D \rangle \)).

Default logic has been applied to databases [9] and the query language DQL based on it was proposed by Cadoli et al. [11].

A default query (DQL Input/Output query) \( Q \) is a pair \( \langle B, D \rangle \), where \( B \) is a set of (function-free and quantifier-free) first-order formulas and \( D \) is a set of open default rules, plus a set \( S = \{ S_1, \ldots, S_n \} \) (output relation schemata) of intensional relations.
Let \( R = \{ R_1, \ldots, R_n \} \) be a database schema, let \( I \) be a database instance over a domain \( \mathcal{U} \) and let \( R_i|I \) be the set of the tuples in \( R_i \). Let \( \zeta \in \mathcal{U}^n \), the instantiations of \( D \) and \( B \) are defined as follows (see [11]):

\[
\text{INST}(D) = \left\{ \begin{array}{l}
\alpha[X/\zeta] : \beta_1[X/\zeta], \ldots, \beta_n[X/\zeta] \\
\gamma[X/\zeta] \alpha(X) : \beta_1(X), \ldots, \beta_n(X) \in D, \zeta \in \mathcal{U}^n
\end{array} \right\}
\]

\[
\text{INST}(B) = \{ \phi[X/\zeta] | \phi(X) \in B, \zeta \in \mathcal{U}^n \}
\]

The completion of the instance \( I \), \( \text{COMP}(I) \), is defined as follows [11]:

\[
\text{COMP}(I) = \{ R_i(a_1, \ldots, a_i) | (a_1, \ldots, a_i) \in (R_i|I) \} \cup \\
\{ -R_i(a_1, \ldots, a_i) | (a_1, \ldots, a_i) \in \mathcal{U}^i \setminus (S_i|I) \}
\]

The default theory with defaults \( \text{INST}(D) \) and first-order formulas \( \text{COMP}(I) \cup \text{INST}(B) \) is: \( Q + I = \langle \text{INST}(D), \text{COMP}(I) \cup \text{INST}(B) \rangle \) [11].

The answer to a default query \( Q \) is the database instance \( J \) for \( S \) defined as follows: under credulous semantics (resp. skeptical/cautious semantics), for each \( S_i \in S \), \( S_i|J \) is the set of all tuples \( t \) such that \( S_i(t) \) is in at least one extension of \( Q + I \) (resp. all the extensions of \( Q + I \)) [11].

4 The view synchronization using the \textit{Compose} and the \textit{Invert} operators

\textit{Generic Model Management} is a recent research area (see, for example, [6]), born to face problems related to the design of metadata management applications and which is based on two main abstractions, schemas and mappings (boxes and arrows in Fig. 1): a schema is an expression that defines a set of possible instances (i.e. database states - \( I \) and \( J \) in Fig. 1); mappings link two schemas describing the relation between the instances of the two schemas themselves (see Fig. 1). Mappings between two schemas \( S \) and \( T \) are described by “set of formulas of some logical formalism over (\( S, T \))” (Fagin et al. [8], p. 999); for example the schema mapping between the two schema versions \( S \) and \( T \) in the example 1 is

\[
\text{map}_{ST} = \{ \forall x, y, z, t(Cities(x,y,z,t) \rightarrow \text{Cities}'(x,y,z)) \}
\]

To solve metadata problems, \textit{Generic Model Management} provides a set of operators allowing to operate both on schemas and mappings. Some of these operators are \textit{Match}, \textit{Compose} and \textit{Invert} (see Fig. 1; for more details and references see [6]).

A way to solve the problem of view synchronization within the \textit{Generic Model Management} is using the inverse operator\(^4\), as described in Fig. 1.

\(^4\) The use of the inverse operator for solving the schema evolution problem can be found in the schema evolution literature in Curino et al. [4] and some notes can be also found in Nash et al. [12].
The solution steps:

1. \( \text{map}_{ST} = \text{Match}(S, T) \)  
   /*find the schema mapping between \( S \) and \( T \)*/

2. \( \text{map}_{TS} = \text{Invert} (\text{map}_{ST}) \)  
   /*"undo" the effects of \( \text{map}_{ST} \)*/

3. \( \text{map}_{RV} = \text{Compose} (\text{map}_{TS}, \text{map}_{SV}) \)  
   /*link the view \( V \) to the new schema \( T' \)*/

Fig. 1. The solution of the view synchronization problem via the \textit{Invert} operator.

In this solution, step 1 realizes the schema change (from \( S \) to \( T \)) through the mapping \( \text{map}_{ST} \), while steps 2 and 3 serve to synchronize the view \( V \). In particular, in step 2 the computation of the inverse is realized by the inverse operator \( \text{Invert} (\text{map}_{ST}) \), which takes the mapping \( \text{map}_{ST} \) between the schemas \( S \) and \( T \) as input and returns a mapping \( \text{map}_{TS} \) between the schemas \( T \) and \( S \). Intuitively, the concept of inverse can be described in the following way: “The inverse operator takes a schema mapping \( \mathcal{M} \) and produces a schema mapping \( \mathcal{M}' \) such that, intuitively, if after applying \( \mathcal{M} \) we then apply \( \mathcal{M}' \), the resulting effect of \( \mathcal{M}' \) is to “undo” the effect of \( \mathcal{M} \.” (Fagin et al. [13], p. 11:2). As many interesting mappings, like the projection, were not invertible, the concept of quasi-inverse has been developed [13], which broadened the set of invertible mappings: for instance, the quasi-inverse of the projection exists and, in the case of the example 1, it is \( \text{Cities'}(x, y, z) \rightarrow \exists t \text{Cities}(x, y, z, t) \). Also the quasi-inverses could be inadequate to realize the “undo” in the context of schema evolution, for the presence of the existential quantifier. In step 3 the composition of the “undo” mapping (\( \text{map}_{TS} \)) and the view \( V \) is linked to the schema \( T \).

5 The idea through examples

In order to extend the solution procedure of Fig. 1 to approximate views, we need a concept of approximate mapping along with two operators, \( \text{DMInvert}(\text{map}_{ST}) \) to realize approximate “undo” and \( \text{DMCCompose}(\text{map}_{TS}, \text{map}_{SV}) \) to compose two approximate mappings.

Consider the mapping \( \text{map}_{ST} \), it is possible to make an approximate “undo” using the following default theory \( \text{DefaultMap}_{TS} \) (we call it \textit{default schema mapping}), obtained applying the following rules: a city with a number of inhabitants greater than two millions and a half \( (N = 2\,500\,000) \), which is not an exception, is a capital and that each country has a unique capital. We denote by \( E_{TS} \) the set \( D \) of defaults and we add the two schema versions \( S \) and \( T \):

\[
T = \{ \text{Cities'} \}
\]
\[ S = \{ \text{Cities} \} \]
\[ \Sigma_{TS} = \begin{cases} \text{Cities}((\text{Name}, \text{Country}, \text{Pop.}) \land \text{Pop.} \geq N : \neg \text{ex(Name)}), \\ \text{Cities}((\text{Name}, \text{Country}, \text{Pop.}, \text{"Yes"}) : \text{ex(Name)}), \\ \text{Cities}((\text{Name}, \text{Country}, \text{Pop.}, \text{"No"}) : \neg \text{ex(Name)}), \\ \text{T} : \neg \text{ex(Name)} \end{cases} \]

The composition \( \text{DMCompose}(\text{DefaultMap}_{TS}, \text{IsCapital}) \) of the default mapping \( \text{DefaultMap}_{TS} \) with the view \( \text{IsCapital} \) in the example 1 produces the following default query, where \( V \) is the set \( S \) of output relations and \( \Sigma_{TV} \) is the set \( D \) of defaults.

\[ V = \{ \text{IsCapital} \} \]
\[ \Sigma_{TV} = \begin{cases} \text{Cities}((\text{Name}, \text{Country}, \text{Pop.}) \land \text{Pop.} \geq N : \neg \text{ex(Name)}), \\ \text{IsCapital}(\text{Name}, \text{Country}) : \text{ex(Name)}), \\ \text{Cities}((\text{Name}, \text{Country}, \text{Pop.}, \text{"Yes"}) : \neg \text{ex(Name)}), \\ \text{T} : \neg \text{ex(Name)} \end{cases} \]

### 6 Default schema mappings

In this section we formalize the concept of default mapping, using that of default query [11].

**Definition 1.** (Default schema mapping) A default schema mapping (DM) is a triple \( (S, T, \Sigma_{ST}) \) where \( S \) (source schema) and \( T \) (target schema) are two relational schemas and \( \Sigma_{ST} \) is a pair \( (B, D) \), where \( B \) is a set of Full T-GDs (see [12]) and \( D \) is a set of open default rules \( \alpha(x); \beta_1(x), \ldots, \beta_n(x) / \gamma(x) \)** such that:
- the prerequisite \( \alpha(x) \) is a conjunction of relations of \( S \) or inequalities;
- the justifications \( \beta_1(x), \ldots, \beta_n(x) (n \geq 0) \) are conjunctions of intensional relations (with negation);
- the consequence \( \gamma(x) \) is either a conjunction of relations of \( T \) or a conjunction of intensional relations (with negation).
The semantics of default mappings is defined following the same steps as for the definition of the semantics of default queries in section 3 (see [11] for details).

We concentrate our attention on credulous semantics, because our work was born for practical motivations (designing a tool to support the work of database administrators during the schema evolution process) and, under credulous semantics, it is possible to provide the database administrator with all the possible solutions.

We adapt the composition algorithms (see [12], p. 15 and [8], pp. 1028-1029) to realize the composition of two DMs.

Algorithm DMC-compose;
input $\mathcal{M}_{ST} = (S, T, \Sigma_{ST})$ and $\mathcal{M}_{TU} = (T, U, \Sigma_{TU})$;
output $\mathcal{M}_{SU} = (S, U, \Sigma_{SU})$;

1. **Initialization step**
   substitute each FullTGDs $\alpha \rightarrow \bigwedge_{i=1}^{n} S_i$ in $\Sigma_{ST}$ with $n$ FullTGDs $\alpha \rightarrow S_i$ substitute each FullTGDs $\alpha \rightarrow \beta$ (both in $\Sigma_{ST}$ and $\Sigma_{TU}$) with the default $\alpha ; \beta$.

2. **Composition step**
   set $\Sigma_{SU} \leftarrow \Sigma_{TU}$;
   repeat
     for all $\delta = \frac{\alpha ; \beta}{\tau} \in \Sigma_{SU}$ do
       for all relations $R(y) \in \alpha$ do
         find all the defaults in $\Sigma_{ST}$ of the form $\frac{\alpha_i ; \beta_i}{\tau_i}$
         if no such defaults exist in $\Sigma_{ST}$ then
           remove $\delta$ from $\Sigma_{SU}$;
         else
           for all $\frac{\alpha_i ; \beta_i}{\tau_i} \in \Sigma_{ST}$ do
             rename its variables like in [12];
             remove $\delta$ from $\Sigma_{SU}$;
             add to $\Sigma_{SU}$ the default obtained from $\delta$ replacing $R(y)$ in $\alpha$ with $\alpha_i$ and adding $\beta_i$ to $\beta$;
           end for
           end if
       end for
     end for
   until every relation symbol in the prerequisite of every formula in $\Sigma_{SU}$ is from $S$.

3. **Construction of $\Sigma_{SU}$**
   add to $\Sigma_{SU}$ all the defaults from $\Sigma_{ST}$ whose consequences are intensional relations.

7 Conclusions and future work

We have presented the concept of default mapping, and we have shown through examples how such mappings can be used within schema evolution context to
reconstruct data and to approximate views. Both theoretical problems (a complete formalization, computational complexity, properties of DMs, etc.) and application ones (how to "discover" default mappings) are still open.

Concept formalization is a very difficult task, because you need to capture relevant, but often not completely known, aspects of the reality you want to model. Formalization within Model Management is also a very complicated task, as shown by the history of recent developments (see [13,8]): mappings and related operators has to be both usable in practical contexts and computationally tractable. Therefore, we have planned to provide a complete formalization only after a period of experimentation of default mappings in real environments, using the framework introduced in this paper. The main problem to face to evaluate if these mappings are actually practical is to devise techniques and algorithms for implementing the DM\textit{invert} operator. To this end, we are working on data mining techniques to build a prototypical tool supporting the db administrator to find the suitable default mappings, when s/he is trying to fix broken views.

References